An Adaptive Methodology for Facial Expression Transfer

Rossana Baptista Queiroz, Adriana Braun and Soraia Raupp Musse
Graduate Program in Computer Science
Pontificia Universidade Catolica do Rio Grande do Sul (PUCRS)
Porto Alegre, Brazil
{*rossanaqueiroz,adriana.braun*}*@acad.pucrs.br, soraia.musse@pucrs.br*

***Abstract*—This work presents a methodology which aims to improve and automate the process of generating facial animation for interactive applications. We propose an adaptive and semiautomatic methodology, which allows to transfer facial expressions from a face mesh to another. The model has three main stages: rigging, expression transfer and animation, where the output meshes can be used as key poses for blendshape-based animation. The input of the model is a face mesh in neutral pose and a set of face data that can be provided from different sources, such as artist crafted meshes and motion capture data. The model generates a set of blendshapes corresponding to the input set, with minimum user intervention. We opted to use a simple rig structure in order to provide a trivial correspondence either with sparse facial feature points based systems or dense geometric data supplied by RGBD based systems. The rig structure can be refined on-the-fly to deal with different input geometric data according to the need. The main contribution of this work is an adaptive methodology which aims to create facial animations with few user intervention and capable or transferring expression details according to the need and/or amount of input data.**

***Keywords*-facial animation; facial rigging; expression transfer**

I. Introduction

In order to animate 3D face meshes, it is necessary to establish control structures that relate their geometry (polygonal mesh) and animation parameters (numerical description of the movements). This process is called *rigging*. Considering the control structures that lead to mesh deformation, we point two main approaches of facial rigging:

1) **using blendshapes**, i.e., a set of key poses (also called morph targets) that can be combined by interpolation with different weights and generates intermediary poses. Usually, the key poses are created manually by an animator using a modeling software. In this case, the control structures are the key poses defined by the animator. These key poses must be different enough from each other in order to reach the space of all the facial expressions we can do, by combining them. The main advantages of this approach are: *i)* the full animators control about the key poses, once they are manually edited; *ii)* its simplicity of implementation; and *iii)* the low computational cost. As cons, we can cite *i)* the need of an artist to prepare each key pose for every different character model; and *ii)* to determine the amount of key poses are essential to generate all the desired facial expressions.
2) **using control structures** (such as feature points, bones, pseudo-muscles,etc), which are responsible by the generation of the mesh deformation, according with their constraints. Usually these structures are defined taking in consideration a parametrization model (such as MPEG-4 [1], FACS [2], etc). The main advantage of this type of rigging is that it does not require the modeling of key expressions. The animator instead should associate the polygonal mesh to the control structures. However, it continues to be a manual step which is time consuming task. As cons, the animator does not have total control over the mesh as it is with the blendshapes, once this control is delegated to the rig control structures. Computational cost is given according to the deformation technique adopted (such as Free Form Deformations [3], Radial Basis Functions [4], [5], etc).

Usually, the rigging is made manually by the animators and it is a time-consuming task. Some work concerning facial animation focus on expression transfer among different face geometries. This is called expression cloning [5], [6] or rigging by example [7], [8]. These work, however, require pre-processing with substantial user intervention in order to associate (register) the meshes. Also, these work use dense correspondence (point cloud/mesh registration) of data from the source to the target mesh. On the other hand, there are some work that use simple control structures to animate the faces [9], [10]. These work require less user intervention in the preparation of the control structure, but the drawback is the loss of details from the source face (because of the simplified control structure).

Contribution. In this context, we propose an adaptive facial expression transfer methodology which aims to create facial animations with few user intervention (without pre-processing) and capable of transferring expression details according to the need and/or amount of input data. Our approach differs from the state-of-the-art papers because it uses a simplified control mask instead dense correspondence

2159-6662/16 $31.00 © 2016 IEEE
DOI 10.1109/SBGames.2015.10

between the source and target meshes. This control mesh can be refined on-the-fly in order to get more details and can generate the animation from a set of blendshapes crafted by an artist or by data provided by images and depth sensors.

II. Related work

Due to the advantages of high mesh control and low computational cost, the use of blendshapes shows an effective way to generate real-time facial animations. As consequence, we can note that several state-of-the-art work have invested in the use of them [11]–[13].

The simpler control structures that lead to mesh deformation are the feature points (FPs), that usually are a set of vertices of the face mesh semantically aligned with a parametrization model, such as MPEG-4 Facial Animation Standard [1] or FACS action units [14]. However, it is necessary to assign for each mesh vertex what control points influence them and how much (weights) they influence. Approaches to handle this require manual intervention or registration methods. Sanchez et al. [9] propose a geometry-based solution to associate the mesh vertices to the control structures. Moreover, instead of considering each feature point independently as a control structure, such as in [10], [15], they propose an approach in which a polygonal mesh created using the MPEG-4 feature points is the surface that control the deformations. Each triangle of this mesh is a control structure that influence the other vertices of the mesh, acting as a planar bone. Their technique reformulates Surface Oriented Free Form Deformation (SOFFDs) to adapt it specifically to face meshes.

Mattos *et al.* [16], [17] and Dutreve *et al.* [18] uses a rig structure similar to ours using feature points-based control. In [16], they describe a facial animation method that uses real three-dimensional models of people, acquired by a 3D scanner. The focus of the work is how to address the correspondence between the scanned point-cloud and the 3D face meshes. They proposed an alternative to the dense correspondence computation by introducing the idea of selecting a sparse set of corresponding points and setting an initial triangulation refined through a subdivision process that matches the intermediate points. Their approach uses structural graph matching to automatically detect the initial set of points, given a 3D face in which the landmarks are previously selected.

The work of Li *et al.* [7] proposes a model to the generation of a set of blendshapes to a neutral face mesh from a set of example blendshapes created previously by an artist. This approach alternates two steps of optimizations in gradient space, in order to better transfer the expression from the source to target face. It can be done iteratively in order to achieve the desired level of refinement. But currently, the algorithm is not fast enough to produce interactive rates and it was the main reason we opted to explore the simplified control mask in our method. In Weise *et al.* [19], Li's model is used in a pre-processing step using the user motion captured data to generate the animations. In Li *et al.* [12], the expression retargeting is done using Laplacian deformation, using RGBD data with correctives on-the-fly.

There are other work that explore ways to automate the association with the control structures, such as the work of Orvalho [20], which automatically transfers a rig defined by an artist to other face meshes. But we opted to explore an approach similar to Sanchez [9], using a control mask, due to three main reasons: *i)* it works in real-time; and *ii)* it is still based on feature points displacements, which is an appropriate approach to handle fast mapping of facial motion capture data provided by Computer Vision algorithms, such as provided by Faceware Live SDK[1]; and *iii)* it has minimal user intervention.

III. Methodology

Figure 1 shows the overall architecture of this model/framework. The model has 3 main stages, which can work separately (using preprocessed inputs) or together, providing a complete facial animation approach:

1) *Rigging Model*: this stage prepares any 3D face mesh for animation. We propose a *Generic Rig Setup* and a *Semiautomatic Rigging* methodology that generates the rig structure. This model requires some user intervention to setup the initial data.
2) *Expression Transfer Model*: this stage transfers the expression data to any rigged face, generating a set of deformed meshes according to the input. Currently we support blendshape-based animation. The facial expressions are generated as key poses, which can be used directly or combined for the generation of the animation frames. The *Blendshape Generation* module is the responsible of transferring the expressions from different sources to the input rigged faces. In this work, we performed expression transfer coming from two sources:
    a) an *Example Set* (key poses already crafted by an artist or edited using our *Animation Control* GUI); and
    b) masks provided by some PDA (Performance-driven Animation) model compliant with our *Generic Rig* approach and generates the *Control Masks Displacements* needed as input to the *Blendshape Generation* model.
3) *Animation Model*: this model allows to create animation sequences using blendshapes. We can specify the animation timeline by scripts either manually edited or provided by other input source. Optionally, we can create additional blendshapes using a GUI, providing a way that allows the user to perform manual adjusts on generated expressions.

[1]Live Driver http://www.image-metrics.com/livedriver

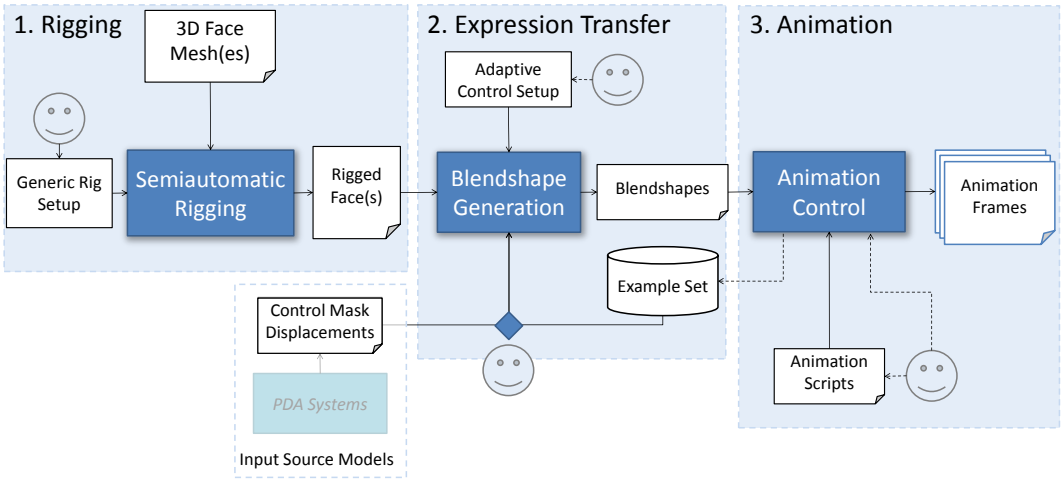


Figure 1: Overall Architecture of the proposed model, showing our 3 main stages/modules.

In order to rig a 3D face mesh, the *Rigging* model (Section IV) requires some input data and user setup procedures, as following:

1) at least one 3D face mesh in neutral pose;
2) to configure the *Generic Rig Setup* , it is necessary to inform (via script or graphical interface) the vertices that will serve as feature points to control the deformations. Optionally, we can provide other data that will personalize the control structure (explained with details in Sections IV-A).

The output is a *Rigged Face* with the vertex associations, as described in Section IV and the RBF regions of influence.

To generate a set of blendshapes using the *Expression Transfer* model (Section V), the input necessary is:

1) at least one *Rigged Face* (3D face mesh + rig data files, describing the *Generic Rig Setup* for such 3D mesh);
2) a set of *Control Mask Displacements* data, which can be provided by different input sources (explained with details in Section V-A);
3) optionally, if we want to refine the Control Mask, we must to provide the number of subdivisions and the triangles/regions of the mask to be subdivided (explained with details in Section V-A1).

The output is a set of 3D face meshes deformed according to the provided *Control Masks Displacements*, which can be used as key poses to a blendshape-based animation model. We called this set of meshes just as *Blendshapes*, once our *Animation* model is blendshape-based. To generate animations, the input necessary is the set of blendshapes and a script describing the timeline (explained with details in Section VI).

IV. Rigging Model

In order to animate 3D face meshes, it is necessary to establish control structures that relate their geometry (polygonal mesh) and animation parameters (numerical description of the movements).This process is a critical step in the animation of faces for Computer Graphics application, since it provides to a face mesh the control structures that allow its animation. Usually, the rigging is made manually by the animators and it is a time-consuming task.

Our approach proposes a simple rig structure in order to provide a trivial correspondence either with sparse facial feature points based systems or dense geometric data supplied by RGBD based systems, we called *Generic Rig Setup*. Using the *Generic Rig Setup*, the *Semiautomatic Rigging* module creates rigged faces with minimum user intervention.

A. Generic Rig Setup

As we mentioned, the *Generic Rig Setup* was planned to be adaptive in order to deal with different input geometric data according to the need. It is composed by the Control Mask and its rig definitions, explained in sequence. Our control structure is inspired by the one from Planar Bones work [9], we called *Control Mask*. In [9], it is a triangle mesh in which the vertices are the feature points specified by the MPEG-4 Facial Animation standard [1]. For our mask, we added 20 feature points provided by Faceware Live[2], intending the integration with the Persona model [21] which uses this development toolkit. Figure 2 shows the *Control Mask*.

The mask has 84 vertices (feature points) and 136 triangles.To setup a *Control Mask* we first have to assign the vertices. Currently, we are doing this step manually but our intention is to integrate with an automated model as future

[2]http://facewaretech.com/products/software/realtime-live/

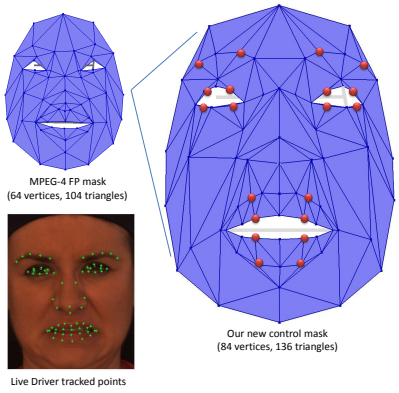


Figure 2: Structure of the *Control Mask*, inspired by [9],which is MPEG-4 compliant (on top of corner left) and the tracked feature points supplied by the Faceware Live development toolkit (on bottom of the corner left).

work (this is the reason the model is not fully automatic). This is the only manual step required to the *Semiautomatic Rigging* model create a rigged face.

As we explain later in Section V-A3, our approach to deform the meshes is done using Radial Basis Functions. To deform a mesh using RBFs, we need to define the set of control points that will lead the displacements, creating their regions of influence (ROIs). In our approach, each triangle of the *Control Mask* define a RBF that deform the vertices associated with it. The control points of each RBF are the vertices from the control triangle and from its adjacent triangles (the triangles which share vertices with the control triangle). Figure 3 shows the generic ROI structure we adopted to this work.

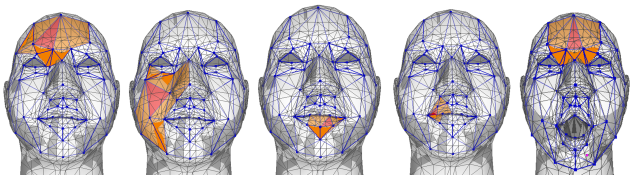


Figure 3: Some RBF ROIs (orange triangles) defined by our model for each selected triangle (red triangles).

Notice that the influence zones can overlap, i.e., most of the triangles of the control mesh belongs to more than one ROI. It means that one FP can influence vertices from different patches. However, each vertice is influenced by just one ROI. We opted to overlap the influence zones in order to produce smoother deformations. This generic ROI structure is computed by the *Semiautomatic Rigging* model, as explained in Section IV-B. Optionally, we can edit ROIs to make adjusts, if we need. Thus, the *Generic Rig Setup* is composed by the *Control Mask* and the generic ROI structure information (we also mention this as *rig definitions*).

B. Semiautomatic Rigging

The *Semiautomatic Rigging* methodology is responsible to generate rigged faces with minimum user intervention. The overall architecture of our model is showed in Figure 4. As input, the user must provide at least one 3D face mesh in neutral pose, as well as the *Generic Rig Setup*. As output, the model generates a *Rigged Face* with the vertex associations and the final RBF ROIs configuration.

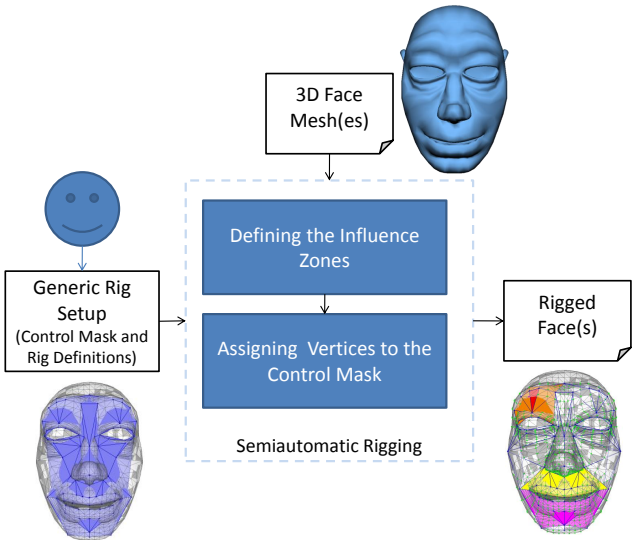
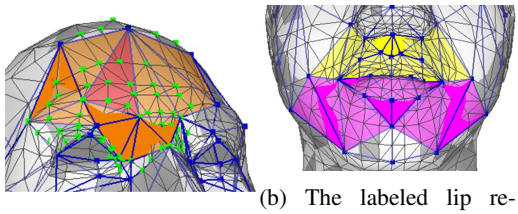


Figure 4: Architecture of the Semiautomatic Rigging model.

The entire rigging pipeline is described in the next sections.

1) Defining the Influence Zones: As described in Section IV-A, the generic ROI structure is composed by a Control Mask triangle and its adjacent neighbors. Figure 5 *a)* illustrates a ROI of the triangle in red, which considers also the vertices from the triangles in orange, that is automatically computed.



(a) Example of a RBF region of influence

(b) The labeled lip regions to separate upper and lower lips.

Figure 5: The RBF influence regions structure.

However, the generic ROIs can not work well with mouth and eyelid regions. Depending of the topology of the 3D mesh, we can generate artifacts on deformation. As an alternative strategy to the use of the texture masks proposed

by Sanchez *et al.* [9] to overcome this problem, our model allows to edit ROIs according to the need.

For eyes and mouth regions, we suggest to edit regions for upper and lower eyelids and lips on the control mesh, which are taken into account at the moment we obtain the adjacent triangles to create the ROIs. Figure 5 *b)* shows the triangles labeled as upper and lower lips. Deformation using RBFs using our ROI structure is explained in details in Section V-A3.

The *Defining the Influence Zones* stage basically receives the *Generic Rig Setup*, adjusts the ROIs according to user input and computes automatically the other ROIs, if needed.

2) Assigning Vertices to the Control Mask: As we mentioned previously, the vertices of the *Control Mask* must be provided by the user. In our prototype, we allow the user to select the vertices either by script or GUI. Then, based on the *Generic Rig Setup*, the generic ROI structure is computed.

Now it is necessary to associate the vertices of the 3D face mesh with the Control Mask. The *Assigning vertices to the Control Mask* stage is important to assure the quality of deformations. Each vertex of the target neutral mesh is associated to one triangle of the control mask, considering the *Generic Rig Setup* (Section IV-A).

As in [9], the triangle mask is the main rig unit. Differently from their approach, our association is fully automated. First, we raycast the mesh vertex normals towards the animation control mask. When there is an intersection, we keep the closer intercepted triangle, if the angle between it and the normal of the intercepted triangle is less than $90°$ (to avoid, for example, to associate lower and upper lips wrongly when there is not labeled regions).

If the vertex does not intercept any triangle or the angle is bigger than $90°$, we associate it with the triangle which has the vertex with the smallest geodesic path (inspired by [22]). Geodesic path is the shortest path between two points on a surface [22]. Considering the face mesh as a graph, we used the BSF (Breadth Seach First) algorithm [23] for computing the geodesic path from the mesh vertices to the Control Mask FPs. We just considered the number of the edges to represent the path, but more accurate solution should be explored as future work [24], [25]. Figure 6 illustrates the geodesic path between vertices and FPs, showing that conectivity helps to find the best association for the vertices.

The association step is required just once to each new mesh we want to animate.

V. Expression Transfer Model

The *Expression Transfer Model* transfers the expression data to any rigged face, generating a set of deformed meshes according to the input. The facial expressions (which also can play the role of microexpressions) are generated as key poses, which can be used directly or combined as blendshapes. In this work, we performed expression transfer using data coming from two different kind of sources:

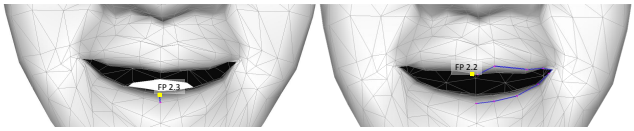


Figure 6: Ilustration of the geodesic path to stablish the correct association between vertices and FPs. Consider the selected vertex (in pink) and the two FPs (3.1 and 3.2, named before MPEG-4). If the mesh is in neutral pose, FPs 2.2(bottom of superior inner lip) and 2.3 (top of inferior inner lip) usually are placed very close (Euclidian distance). So, if we consider just the Euclidian distance between the selected vertex and these FPs, it can be errouneously associated with the 2.2. However, if we consider the conectivity using geodesic distance, we clearly see that FP 2.3 is the correct one.

1) an *Example Set* (key poses already crafted by an artist or edited using our *Animation Control* GUI); and
2) masks provided by PDA models which is fully compliant with our *Generic Rig* approach and generates the *Control Masks Displacements* needed as input to the *Blendshape Generation* model.

The *Blendshape Generation* module is the responsible of transferring the expressions from different sources to the input rigged faces, as we describe in the next section.

A. Blendshape Generation

Figure 7 shows the architecture of the *Blendshape Generation* module. In the current stage of this work, the *Expression Transfer Model* always generate, as output, a set of blendshapes (deformed meshes). As we chose to work with blendshape-based animation, our *Animation Module* just performs the blendshape interpolation to create the final frames. Each stage from Figure 7 is explained in the next sections.

1) Adaptive Subdivsion: With the objective of obtaining more details from the source mesh, we propose an *Adaptive Subdivision* stage, inspired by Mattos *et al.* work [17]. As mentioned previously, we can point two main reasons to propose this approach:

1) to deal with different input source data coming from different acquisition systems, such as feature points, point clouds and other meshes, taking advantage of using different amount of information provided by them;
2) to avoid the use of all dense data provided by RGBD systems or meshes which topologies are not compatible with real-time applications.

It is import to say that, at the current stage of this work, we are not working with RGBD data. For now, we used this methodology only in 3D meshes, but it also possible to

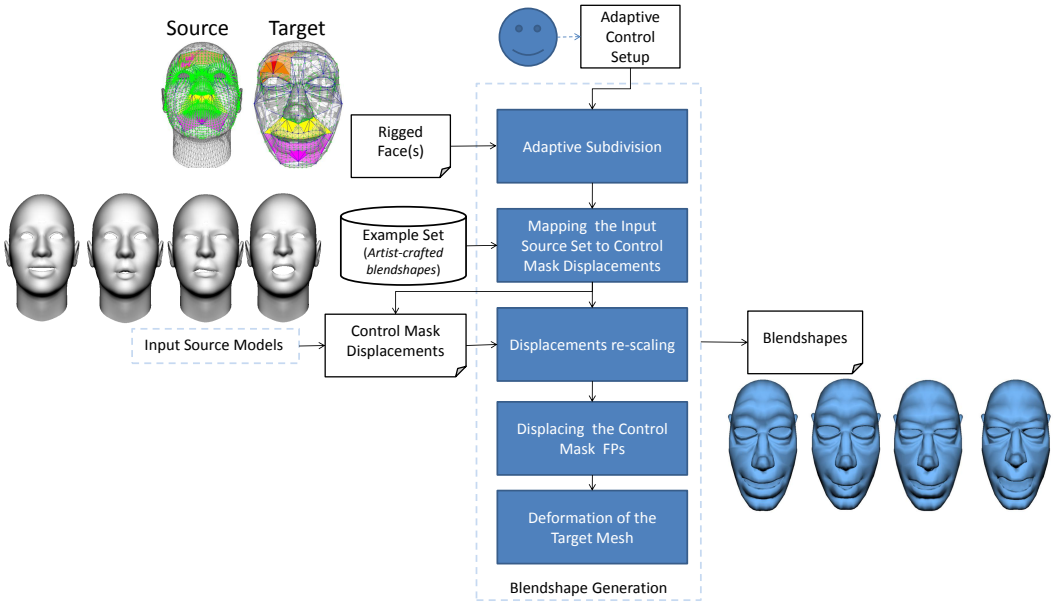


Figure 7: Architecture of the Blendshape Generation model.

use point clouds, if the Control Mask feature points were accordingly labeled.

The *Adaptive Subdivision* is performed using the *Control Mask*. The user must to specify the number of subdivisions and optionally the triangles he/she wants to be subdivided. This data is called *Adaptive Control Setup*. If the user does not configure this setup, there are no subdivision and the original Control Mask is used in the another stages of the pipeline. Also, if user specifies only the number of subdivisions, all triangles of the Control Mask will be subdivided, as default.

This stage works by creating new triangles from the initial ones and adding the new vertices as control points to the RBFs ROIs of these triangles. We can subdivide recursively the new triangles according to the need, i.e., the number of subdivisions defined by the user. Figure 8 illustrates the subdivision process. The new vertices from the face mesh are chosen according to their Euclidean distance to the computed triangle centroid located on control mask. Figure 9 illustrates a Control Mask after one and two recursive subdivisions.

By adding more triangles to the control mask, we can preserve more details from the *Example Set* (if it is composed by 3D meshes or point clouds).

2) Mapping the Input Source Set to Control Mask Displacements, Displacements scaling and Displacing the Control Mask FPs: The next three steps of the *Blendshape Generation* model make the mapping between the input Control Mask Displacements and the target mesh (the 3D face mesh which the expressions will be transferred):

- *Mapping the Input Source Set to Control Mask Displacements*: considering that the Example Set has vertex correspondence, so we have a set $B = \{b_0, b_1, ...b_n\}$ of n control masks (b_0 is the mesh in neutral pose). For each b_i, we get the displacement

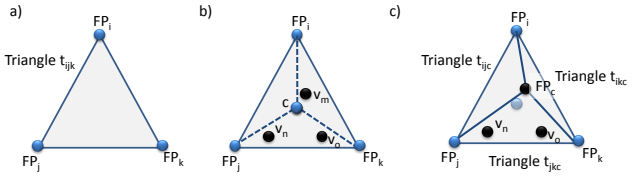


Figure 8: Steps of mask subdivision: a) Control Mask triangle composed by the FPs i, j and k; b) the centroid c of the triangle is computed; c) for each vertice v associated to the triangle, it is computed the Euclidian distance from v to c, and the closer vertice is chosen (notice that the computed centroid is a point located on the triangle plane, and not a vertex of the face mesh).

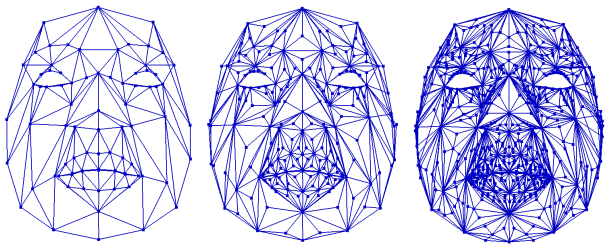


Figure 9: Example of Control Mask subdivision. From left to right: original, after 1 and 2 subdivisions.

vector $\vec{d_i}$ from b_i to b_0 ($\vec{d_i} = b_i - b_0$).

- *Displacements scaling*: in this step, we scale the set of displacements $D = \{d_0, d_1, ...d_n\}$ using the MPEG-4 FAPU of the source face $sFAPU$. These key distances must be computed to each face mesh, and it assures a coherent transference of the FPs displacements among different face topologies. The displacement is divided by FAPU, according to the FP is corresponding to d_i, as Equation 1 shows:

$$s\vec{d}_i = \vec{d}_i / sFAPU. \quad (1)$$

- *Displacing the Control Mask FPs*: using FAPU scaling, it is trivial to displace the vertices of the target Control Mask, just rescaling to its FAPU. In this stage, the scaled displacement $s\vec{d}_i$ is now scaled to the FAPU of the target mesh $tFAPU$. Then, the new of the FP_i is given by Equation 2:

$$b'_i = b_0 + s\vec{d}_i tFAPU. \quad (2)$$

3) Deformation of the target mesh: The last stage of the model is the *Deformation of the target mesh*, which deforms the target mesh and generates the blendshapes with the input expressions. In this work, we used the approach developed by Noh *et al.* [4] (which proposes the animation of faces using RBFs), and also the methodology presented by Dutreve *et al.* [10] (who uses the RBFs to map the real and virtual parameter spaces). In both articles, the multi-quadratic RBF funcion is used. Each RBF ROI is defined considering the triangles neighborhood as described previously (Figure 10 shows an example of a RBF ROI of a rigged face).

Let be $S = (\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_N)$ the source FP set (corresponding to the neutral face) of a ROI. The first step is to train the RBF using this set. Then, for each target FP set (corresponding to the displaced FPs) $T = (\vec{t}_1, \vec{t}_2, \ldots, \vec{t}_N)$, the set of weights $W = (w_1, w_2, \ldots, w_N)$ is obtained so that, when applied to the multi-quadratics RBF $F(\vec{s}_j)$ of all the other points j of the ROI, will determine their new position ($\vec{t}_j = F(\vec{s}_j)$). Each one of these steps is briefly described below [10]:

- **Training**: to train the RBF, it is necessary to determine the matrix H, which contains the multi-quadrics function applied to all pairs of elements of S, according to the Equation 3

$$H_{ij} = h(\|\vec{s}_j - \vec{s}_i\|) = \sqrt{(\|\vec{s}_j - \vec{s}_i\|)^2 + sc_j^2}, \quad (3)$$

where $sc_j^2 = min_{j \neq i}(\|\vec{s}_j - \vec{s}_i\|)$ is a stiffness coefficient sugested by Eck [26] to smooth the deformations where the FPs are sparse and to reinforce where they are very close.

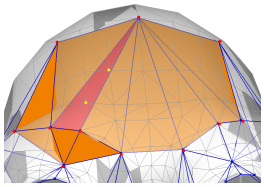


Figure 10: Example of a RBF ROI, corresponding to the triangle in red. Yellow points are the vertices associated to the triangle. The orange triangles are the neighborhood of the red triangle, which compose its ROI. The yellow points are influenced by the red points.

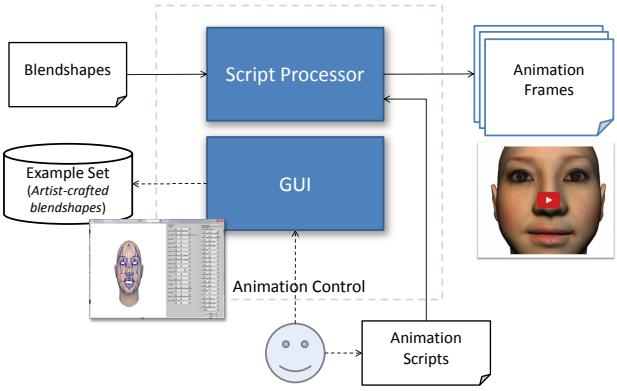


Figure 11: Overview of the Animation Control module.

- **Obtaining the weight vector**: let be $T_x = (t_1^x, t_2^x, \ldots, t_N^x)$ the set with the new positions of the FPs (t_j^x is the x coordinate of the $\vec{t}_j$), we obtain the weight vector for the x axis (and also for the y and z in the same way), as shown in the Equation 4:

$$T_x = H \cdot W_x W_x = H^{-1} T_x \quad (4)$$

- **Obtaining the new positions**: once the RBF has been trained for each axis, the new position for each point $\vec{t}_j$ of each $\vec{s}_j$ of the ROI is obtained through the function $F(\vec{s}_j)$, as shown in the Equation 5:

$$\vec{t}_j = F(\vec{s}_j) = \sum_{i=1}^{N} \vec{w}_i \cdot h(\|\vec{s}_j - \vec{s}_i\|) \quad (5)$$

VI. Animation Model

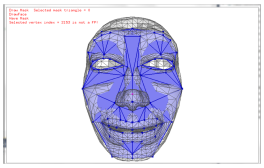
The *Animation Model* is the last stage of our animation model. Overal architecture of the *Animation Control* is showed in Figure 11. We can specify the animation timeline by *Animation Scripts* either manually edited or provided by other input sources (such as Braun [21]). Optionally, we can create additional blendshapes using a GUI, providing a way that allows the user to perform manual adjusts on generated expressions.

Basically, these scripts inform the durations of each expression and the blendshapes weights to compose it. So, each line of the script contains the duration followed by a stream of weights. We perform linear interpolation between the expressions, according to the specified durations. The scripts are processed by the *Script Processor* stage, which interpolates the blendshapes and generate the *Animation Frames*.

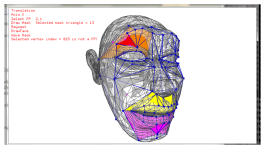
Finally, we added a simple GUI that allows the user to make adjusts on the generated blendshapes or just create new key poses combining blendshape weights and saving them to be used for generating animations or being part of an *Example Set* (transferred to another face meshes using the *Expression Transfer* model.

VII. Results

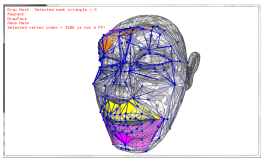
The prototype was developed using the C++ programming language and OpenGL/Glut as graphic API (Application Programming Interface). Generally speaking, the prototype provides a GUI for the 5 kinds of user intervention the framework requests (see the user faces illustrated in Figure 12) by implementing two pieces of software (we called *Rigging and Expression Transfer (R&T)* and *Animator* tools):



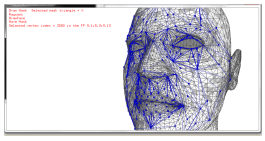
(a) Edit/Debug Mode for the Control Mask



(b) Edit/Debug Mode for the RBF ROIs



(c) Edit/Debug Mode for the Adaptive Control Setup



(d) Edit/Debug Mode for the Adaptive Control Setup (with another face)

Figure 13: Four screenshots showing the prototype GUI for Rigging and Expression Transfer.

1) for the *Generic Rig Setup* configuration, as explained in Section IV (illustrated in Figure 13 a) and b)), the R&T GUI provides an interface for clicking on the vertices of the mesh that we want to define as the control mask feature points and also to configure the ROIs, if the user wants to use personalized ROIs, as discussed in Section IV-A. The user can add or delete triangles of a neighborhood and also label semantic regions (for example, the upper and bottom lips regions showed previously in Figure 5 b));

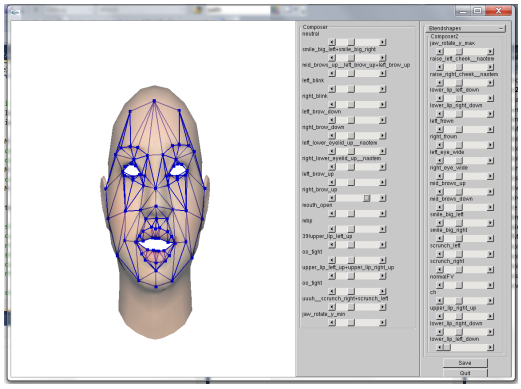


Figure 14: Screenshot showing the prototype GUI for blendshape manual composition and generation of the animations (by loading scripts).

2) for the *Adaptive Control Setup* configuration (this is an optional step, as described in Section V (illustrated in Figure 13 c) and d)). This is also done using the R&T GUI, where the user can specify the number of subdivisions and optionally select the triangles (specific areas) he/she wants to be subdivided;
3) for defining the input data source to be used by the *Blendshape Generation* module, wich can be either a set of existing blendshapes with vertex correspondence (*Example Set*);
4) for loading the *Animation Scripts*, using the Animator GUI;
5) for composing and adjust blendshapes (illustrated in Figure 14), also described in Section VI, through a slider-based interface, where the user can combine a set of key poses and save both the mesh and weights (for using in within the animation scripts) into the Animator GUI.

Using the R&T and Animator tools, we present some obtained results from our model.

A. Transferring Expressions

The *Expression Transfer* is the methodology we proposed to create personalized facial expressions to an arbitrary target 3D face mesh. To achieve this goal using our prototype, we should perform the following steps:

1) To perform the *rigging* of both source and target face meshes. The user should specify the 84 FPs according to the *Control Mask* topology presented in Section IV. Once specified the *Control Mask* vertices, the regions of influence can be optionally redefined (by selecting the neighborhood of a specific triangle). If the user does not redefine the ROIs, the standard ROI for each triangle is its adjacent neighbors (as described in Section IV). After this, it is performed the mapping of all the other vertices of the input 3D face mesh,

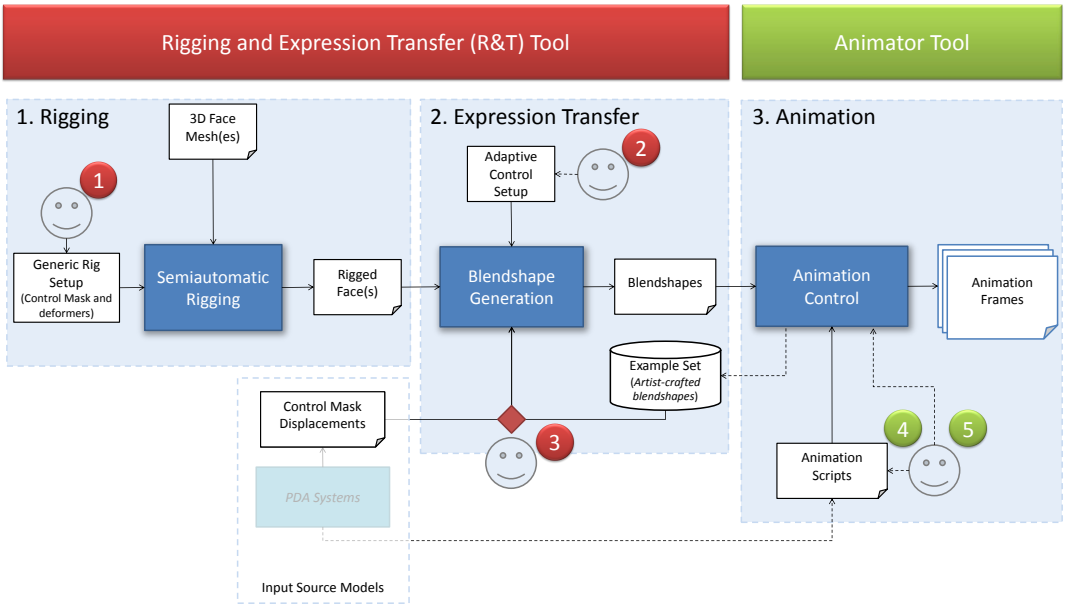


Figure 12: Scheme showing the covering of the visual tools into the proposed model, considering the required user intervention (enumerated from 1 to 5)

described in Section IV-B as the *Assigning Vertices to the Control Mask* step. Then, all these data are saved and the input 3D face mesh is rigged; and

2) To generate the blendshapes (as result of the *Expression Transfer*). To achive this, the user should specify a *Rigged Face*, the input data source (mask displacements or the blendshapes example set) and optionally the *Adaptive Control Setup*, defining the number of subdivisions and the triangles to be subdivided. Then, a set of blendshapes with the expressions from the input data source transferred to the target face are generated. Both source and target face meshes must be rigged previously. Figure 15 illustrates the results obtained by the transferring of an 8 facial expressions set (generated previously using some FaceGen's basic expressions).

Figures 16 to 19 show some results obtained by the *Expression Transfer* model using the FaceGen's meshes as being the input *Example Set* using target meshes with different topologies and our default *Control Mask*, without subdivisions.

Visual results shows the viability of the proposed methodology. By visual inspection, we can notice that:

1) in general, the *Control Mask* without subdivisions works "well"(in the sense of produce visually satisfactory results) for the 3D meshes we tested. These 3D models are designed to be used in real-time applications, even the high-polygonal set, so these meshes does not present small wrinkles, bulges and furrows which could demand an special treatment

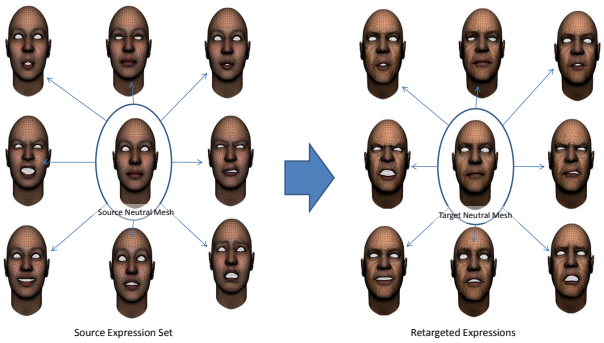


Figure 15: Some results obtained using our method. From the example set (left) and the target neutral face (central face at right), our model generates the retargeted expressions (faces surrounding the target neutral face) by deforming the target neutral face.

2) some results (such as Figures 17 and 19) present some artifacts in the chin contour area. This occurs because the vertices closer to the mask area should be also associated to the closer mask triangle. To overcome this problem is necessary to stablish a treshold, which could be, for example, supplied by using the geodesic distance (considering connectivity to chose the contour nearby vertices).

We also got some results from exploring the *Adaptive Subdivision* module, when using one, two and four recursive subdivisions. Figure 20 plots the normalized distance errors of generated mesh vertices from transferring a expression

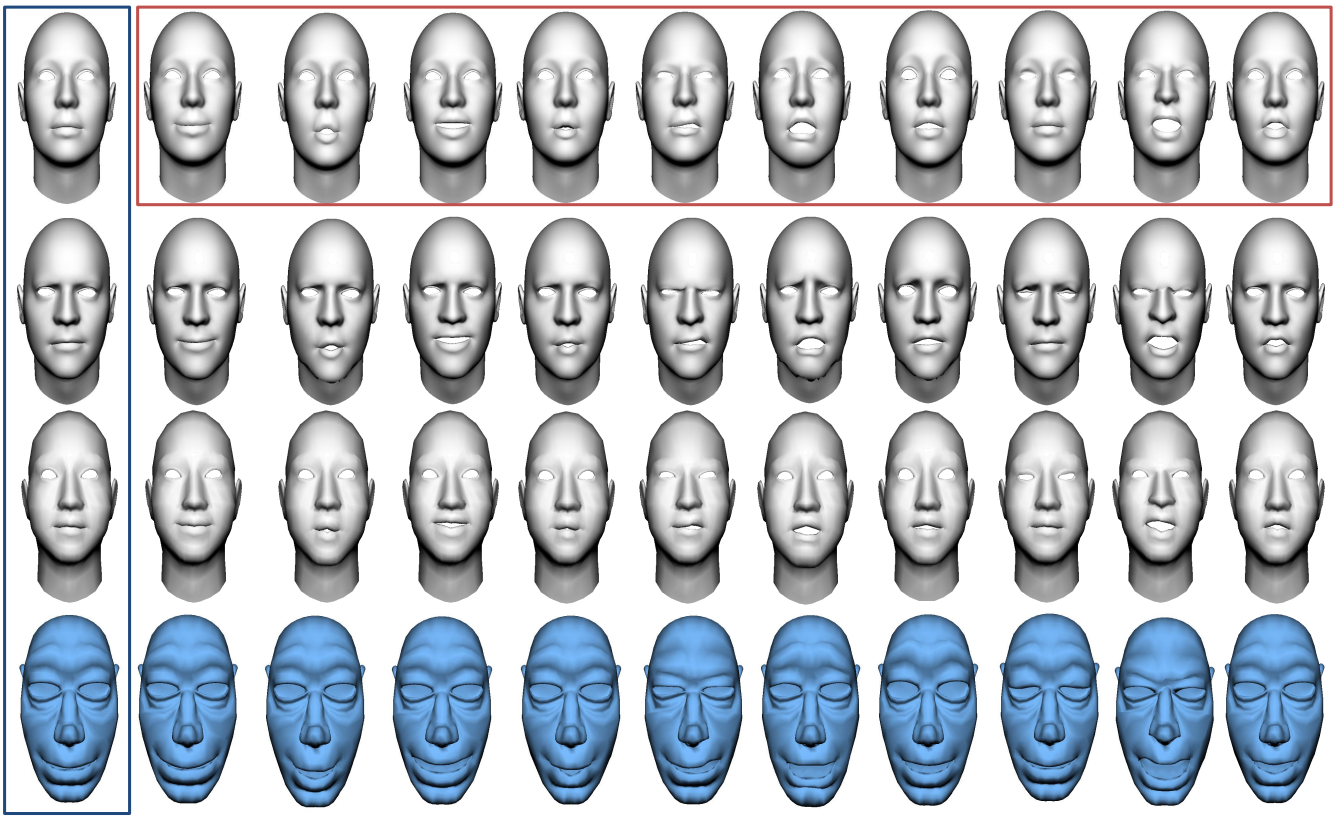


Figure 16: Some results of our model. The first row presents the expressions from the example set (neutral face and the set of facial expressions inside the red box). Also, user must supply target face meshes in neutral pose (faces inside the blue box). Rows/lines 2 to 4 show generated blendshapes for four different target faces.

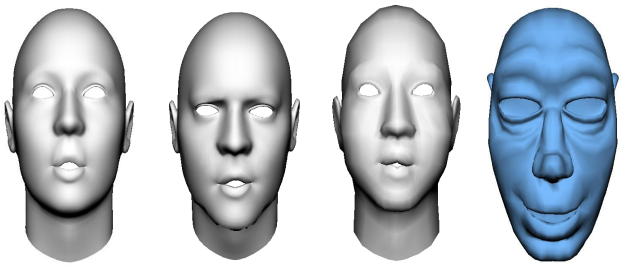


Figure 17: A close of the ”kiss mouth” expression transfer among different face topologies. The first face is the source, the three remaining are the resulting meshes.

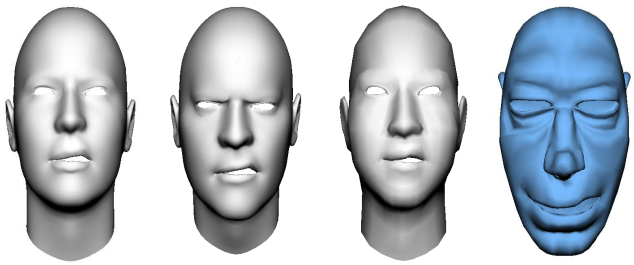


Figure 18: A close of the disgust expression transfer among different face topologies. The first face is the source, the three remaining are the resulting meshes.

using the same mesh as source and target. To compute these errors, we calculated the difference from each vertex of the generated mesh to the correspondent vertex on the source mesh. Then, we scaled these differences by the biggest difference found. It was done just for helping in a preliminary visual analysis. We considered to compare the meshes using some point-cloud comparing metric, such as the Hausdorff Distance [27], but preliminary tests showed no significant differences among the meshes considering all the face mesh vertices. As future work, we intend to investigate a better metric to perform a reliable quantitative evaluation.

We can notice that, increasing the number of subdivisions, we can preserve more details from the source mesh, and also our deformation using RBFs still producing smooth surfaces. This indicates that our adopted methodology for subdivision using our *Generig Rigging Setup* (control mask and ROIs)

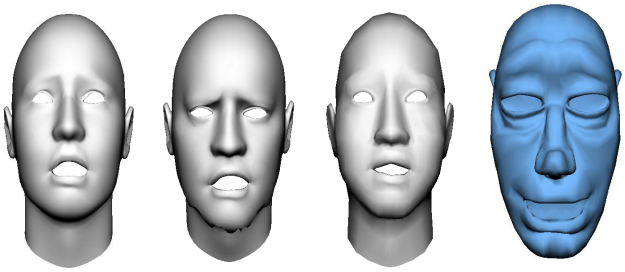


Figure 19: A close of the fear expression transfer among different face topologies. The first face is the source, the three remaining are the resulting meshes.

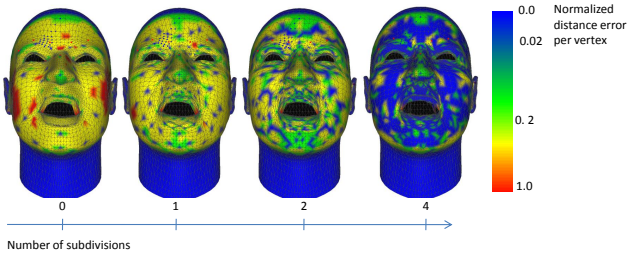


Figure 20: Difference of each correspondent vertex distances from source to the generated mesh using zero, one, two and four recursive subdivisions on each triangle of the mesh.

definitions is suitable for expression transfer with different levels of detail.

If the user wants to generate an animation using the generated blendshapes, he/she should specify an *Animation Script* to the *Animator Tool*. Alternatively, a script with a set of 3D face meshes (with vertex correspondence) can be selected and the Animator Tool supplies a GUI with sliders to manually combine and generate new key poses.

Generally speaking, we can see that the results show the viability of the proposed methodology, and the refinement of the control mask contributes to preserve more details from the source mesh. As immediate future work, we intend to explore ways to improve the mask refinement step, providing clues of the best triangles or regions of the mask should be subdivided according to the source expression.

VIII. Final Remarks

This work presented a methodology for generating blendshape-based facial animation in arbitrary 3D face meshes. Our method is capable of transferring the main facial features of the expression performed by an individual through a framework which deals with a simple but adaptive control structure and different input data sources.

Recalling the main contributions of this work, this present work provided:

1) A semiautomatic model for rigging 3D faces, based on a flexible and adaptive rig structure we called *Control Mask*. In despite of models which perform dense correspondence among source and target facial data, which can not be done in real-time, we proposed an adaptive rig structure which can be refined according to the user/application's need. Our control mask was designed based on two premises:
   a) to keep feature points with a semantics as simple as possible for human understanding (for example, tip of the nose), based on MPEG-4 FA standard, to ease the process of rigging by the users
   b) to provide a trivial correspondence with the state-of-the-art real-time facial motion capture tools, such as the Faceware Live[3].
2) A model that can generate blendshapes from different input sources, such as another meshes or PDA data: as mentioned previously, the idea of adaptivity to different input sources was always present in the model's design, considering to reproduce facial expressions either crafted by artists or captured by users.

Concerning to our proposed *Rigging Model*, we can state that, although it does not present a noticeable novelty compared to the state-of-the art, the results obtained by the *Semiautomatic Rigging* module produces a robust rigged faces with few user intervention. The proposed methodology works with both low and high polygonal 3D mesh topologies (designed for real-time applications) and with different representation languages (for example, with our cartoon-like blue ogre).

The proposed RBFs ROIs scheme was designed trying to simplify the rigging process, considering the Control Mask FPs topology as the starting point. The possibility of labeling semantic regions should be more deeply investigated, in order to suggest if there is an "optimal ROI topology" (such as in). Another utility for the semantic regions can be the creation of semantic layers, as in Kholgade *et al.* [28], where the performance of the captured actor is decomposed in 3 different layers (emotion, speech and blink) activating specific facial regions.

As future work, we can cite:

- Full automation of the *Rigging Model*, providing an automatic FP estimation at the beginning of the rigging process, aiming to supply the FPs coordinates of an arbitrary input face mesh, using some state-of-the-art automatic landmark detection of 3D meshes technique [29].
- Study and proposition of a methodology aiming to improve our FAPU-based mapping. We observed that this mapping, based on normalizing the facial motion displacement vector by the key-distances proposed by the MPEG-4 FA standard should be improved in the

[3]http://facewaretech.com/products/software/realtime-live/

sense of trying to find a way to fit better the displacements across different face topologies. We notice that although the traditional mapping using RBFs, used in several work [6], [18], [20], as well as our methodology, can generate robust results, but it sometimes seems to conceal the target face features. We should start by performing a comparative study among mapping techniques, and then try to propose a model aiming to personalize the target face without uncharacterizing this.
- Study and proposition of metrics for detecting the regions of the which more contributes to the expressiveness, in order to select automatically the control mask triangles to be subdivided. We started a preliminary study using the modified Hausdorff Distance [27] for face regions, but we did not obtain results to be showed in the scope of this work;
- Adding an expression wrinkles generation layer, to allow the expression personalization considering the smaller skin deformations, according to the need of the application (but always focusing on real-time deformation).

ACKNOWLEDGMENT

The authors would like to acknowledge CAPES and CNPq for the funding support.

REFERENCES

[1] I. S. Pandzic and R. Forchheimer, Eds., *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[2] P. Ekman, W. V. Friesen, and J. C. Hager, "Facial action coding system – the manual," Sample available in http:http://face-and-emotion.com/dataface/facs/manual/TitlePage.html, 2002.

[3] P. Kalra, A. Mangili, N. Magnenat-thalmann, and D. Thalmann, "Smile: A multilayered facial animation system," in *In T.L. Kunii, editor, Modeling in Computer Graphics*. Springer-Verlag, 1991, pp. 189–198.

[4] J. yong Noh, D. Fidaleo, and U. Neumann, "Animated deformations with radial basis functions," in *VRST '00: Proceedings of the ACM symposium on Virtual reality software and technology*. New York, NY, USA: ACM, 2000, pp. 166–174.

[5] J.-y. Noh and U. Neumann, "Expression cloning," in *ACM SIGGRAPH 2006 Courses*, ser. SIGGRAPH '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1185657.1185862

[6] Y. Seol, J. Lewis, J. Seo, B. Choi, K. Anjyo, and J. Noh, "Spacetime expression cloning for blendshapes," *ACM Trans. Graph.*, vol. 31, no. 2, pp. 14:1–14:12, Apr. 2012. [Online]. Available: http://doi.acm.org/10.1145/2159516.2159519

[7] H. Li, T. Weise, and M. Pauly, "Example-based facial rigging," *ACM Trans. Graph.*, vol. 29, pp. 32:1–32:6, July 2010. [Online]. Available: http://doi.acm.org/10.1145/1778765.1778769

[8] C. Cao, Y. Weng, S. Lin, and K. Zhou, "3d shape regression for real-time facial animation," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 41:1–41:10, Jul. 2013.

[9] M. Sanchez and S. Maddock, "Planar bones for mpeg-4 facial animation," in *TPCG '03: Proceedings of the Theory and Practice of Computer Graphics 2003*. Washington, DC, USA: IEEE Computer Society, 2003, p. 81.

[10] L. Dutreve, A. Meyer, and S. Bouakaz, "Feature points based facial animation retargeting," in *VRST '08: Proceedings of the 2008 ACM symposium on Virtual reality software and technology*. New York, NY, USA: ACM, 2008, pp. 197–200.

[11] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. Pighin, and Z. Deng, "Practice and Theory of Blendshape Facial Models," in *Eurographics 2014 - State of the Art Reports*, S. Lefebvre and M. Spagnuolo, Eds. The Eurographics Association, 2014.

[12] H. Li, J. Yu, Y. Ye, and C. Bregler, "Realtime facial animation with on-the-fly correctives," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 42:1–42:10, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2462019

[13] F. Xu, J. Chai, Y. Liu, and X. Tong, "Controllable high-fidelity facial performance transfer," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 42:1–42:11, Jul. 2014.

[14] P. Ekman and W. Friesen, *Facial Action Code System*. Palo Alto, CA: Consulting Psychologists Press, Inc., 1978.

[15] R. B. Queiroz, A. Braun, J. L. Moreira, M. Cohen, S. R. Musse, M. R. Thielo, and R. Samadani, "Reflecting user faces in avatars," in *Proceedings of the 10th international conference on Intelligent virtual agents*, ser. IVA'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 420–426. [Online]. Available: http://dl.acm.org/citation.cfm?id=1889075.1889127

[16] A. B. Mattos and R. M. Cesar Jr, "3d facial animation based on structural registration," in *Workshops of Sibgrapi 2009-Posters*, 2009.

[17] A. B. Mattos, J. P. Mena-Chalco, R. M. Cesar Jr, and L. Velho, "3d linear facial animation based on real data," in *Graphics, Patterns and Images (SIBGRAPI), 2010 23rd SIBGRAPI Conference on*. IEEE, 2010, pp. 271–278.

[18] L. Dutreve, A. Meyer, V. Orvalho, and S. Bouakaz, "Easy rigging of face by automatic registration and transfer of skinning parameters," in *Proceedings of the 2010 International Conference on Computer Vision and Graphics: Part I*, ser. ICCVG'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 333–341. [Online]. Available: http://dl.acm.org/citation.cfm?id=1882208.1882249

[19] T. Weise, S. Bouaziz, H. Li, and M. Pauly, "Realtime performance-based facial animation," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 77:1–77:10, Jul. 2011. [Online]. Available: http://doi.acm.org/10.1145/2010324.1964972

[20] V. C. Orvalho, E. Zacur, and A. Susin, "Transferring the rig and animations from a character to different face models," *Computer Graphics Forum*, vol. 27, no. 8, pp. 1997–2012, 2008. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2008.01187.x

[21] A. Braun, "Aprendizado e utilizao do estilo de movimento facial na animao de avatares," 2014.

[22] X. Wan, S. Liu, J. Chen, and X. Jin, "Geodesic distance-based realistic facial animation using rbf interpolation," *Computing in Science Engineering*, vol. 14, no. 5, pp. 49–55, Sept 2012.

[23] C. Lee, "An algorithm for path connections and its applications," *Electronic Computers, IRE Transactions on*, vol. EC-10, no. 3, pp. 346–365, Sept 1961.

[24] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe, "Fast exact and approximate geodesics on meshes," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 553–560, Jul. 2005. [Online]. Available: http://doi.acm.org/10.1145/1073204.1073228

[25] D. Martínez, L. Velho, and P. C. Carvalho, "Computing geodesics on triangular meshes," *Computers & Graphics*, vol. 29, no. 5, pp. 667–675, 2005.

[26] M. Eck, "Interpolation methods for reconstruction of 3d surfaces from sequences of planar slices," *CAD und Computergraphik*, vol. 13, no. 5, pp. 109–120, 1991.

[27] M. Hossain, M. Dewan, K. Ahn, and O. Chae, "A linear time algorithm of computing hausdorff distance for content-based image analysis," *Circuits, Systems, and Signal Processing*, vol. 31, pp. 389–399, 2012, 10.1007/s00034-011-9284-y. [Online]. Available: http://dx.doi.org/10.1007/s00034-011-9284-y

[28] N. Kholgade, I. Matthews, and Y. Sheikh, "Content retargeting using parameter-parallel facial layers," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '11. New York, NY, USA: ACM, 2011, pp. 195–204. [Online]. Available: http://doi.acm.org/10.1145/2019406.2019433

[29] C. Creusot, N. Pears, and J. Austin, "A machine-learning approach to keypoint detection and landmarking on 3d meshes," *Int. J. Comput. Vision*, vol. 102, no. 1-3, pp. 146–179, Mar. 2013. [Online]. Available: http://dx.doi.org/10.1007/s11263-012-0605-9